

Listing of Claims

1. (Currently Amended) An inter-module interface definition A computer-readable medium comprising:

a command definition, wherein

 said command definition comprises commands for interfacing with a multi-channel, multi-media, communication queuing system, and
 said commands are independent of a media type of a communication channel
 of the multi-channel, multi-media, communication queuing system;
 and
 instructions to use at least one of the commands of the command definition to
 support communication via the communication channel of the multi-channel,
 multi-media, communication queuing system.

2. (Currently Amended) The inter-module interface definition computer-readable medium of claim 1, wherein the command definition includes at least one of the following channel driver commands to: request media type lists and command event lists, create driver objects, request service objects, and release driver objects.

3. (Currently Amended) The inter-module interface definition computer-readable medium of claim 1, wherein the command definition includes at least one of the following service object commands to: release service objects, issue a notice when handling of an event is complete, invoke commands, release work items, suspend work items, resume work items, handle queued events, and cancel queued events.

4. (Currently Amended) The inter-module interface definition computer-readable medium of claim 1, wherein the command definition includes at least one of the following client object commands to: start a work item, release work items, save work item contexts, restore work item contexts, serialize work items, free work item storage, begin batch processing, and end batch processing.

5. **(Currently Amended)** A method of inter-module communication comprising:
defining using a command definition to communicate via a multi-channel, multi-
media, communication queuing system, wherein
said command definition comprises commands for interfacing with [[a]] the
multi-channel, multi-media, communication queuing system, and
said commands are independent of a media type of a communication channel
of the multi-channel, multi-media, communication queuing system.

6. **(Currently Amended)** The method of claim 5 further comprising defining using
at least one of the following channel driver commands for: requesting media type lists and
command event lists, creating driver objects, requesting service objects, and releasing driver
objects.

7. **(Currently Amended)** The method of claim 5 further comprising defining using
at least one of the following service object commands for: releasing service objects, issuing a
notice when handling of an event is complete, invoking commands, releasing work items,
suspending work items, resuming work items, handling queued events, and cancelling queued
events.

8. **(Currently Amended)** The method of claim 5 further comprising defining using
at least one of the following client object commands for: starting a work item, releasing work
items, saving work item contexts, restoring work item contexts, serializing work items, freeing
work item storage, beginning batch processing, and ending batch processing.

9. **(Original)** A computer readable storage media comprising:
computer instructions to implement the method of claim 5.

10. **(Original)** A signal in a carrier medium comprising:
computer instructions to implement the method of claim 5.

11. (Currently Amended) A communication server comprising:
instructions configured to support communication via a communication channel,
wherein
the communication channel communicates via a type of a plurality of types of
communication media,
the instructions conform to an interface command definition comprising
commands for interfacing with one or more communication channel
drivers for the plurality of multiple types of communication media, and
the commands are independent of the type of the communication media of
the communication channel.

12. (Previously Presented) The communication server of claim 11, wherein the command definition includes a command to start a work item.

13. (Previously Presented) The communication server of claim 11, wherein the command definition includes a command to release a work item.

14. (Previously Presented) The communication server of claim 11, wherein the command definition includes a command to save a work item context.

15. (Previously Presented) The communication server of claim 11, wherein the command definition includes a command to restore a work item context.

16. (Previously Presented) The communication server of claim 11, wherein the command definition includes a command to serialize a work item.

17. (Previously Presented) The communication server of claim 11, wherein the command definition includes a command to free work item storage.

18. (Previously Presented) The communication server of claim 11, wherein the command definition includes a command to begin batch processing.

19. (Previously Presented) The communication server of claim 11, wherein the command definition includes a command to end batch processing.

20. (Previously Presented) The communication server of claim 11, further comprising:

a client object operable to interface with the one or more communication channel drivers using at least a portion of the command definition.

21. (Previously Presented) The communication server of claim 11, further comprising:

a plurality of client objects, wherein each client object interfaces with a service object in one of the communication channel drivers, wherein each service object and each client object correspond to one type of communication media.

22. (Currently Amended) A channel driver comprising:

instructions configured to receive a command that conforms to an interface command definition comprising commands for interfacing with a multi-channel, multi-media, communication queuing system, wherein
the commands are independent of a media type of a communication channel
of the multi-channel, multi-media, communication queuing system;

and

instructions configured to issue the commands to the communication channel.

23. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to request a media type list.

24. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to request a command event list.

25. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to create a driver object.

26. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to request a service object.

27. (Previously Presented) The channel driver object of claim 22, wherein the command definition includes a command to release a driver object.

28. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to issue a notice when handling of an event is complete.

29. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to invoke commands.

30. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to suspend work items.

31. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to resume work items.

32. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to handle queued events.

33. (Previously Presented) The channel driver of claim 22, wherein the command definition includes a command to cancel queued events.

34. (Previously Presented) The channel driver of claim 22, wherein the channel driver is operable to interface with a communication server and at least one communication device.

35. (Previously Presented) The channel driver of claim 34, further wherein the communication server is operable to interface with a queuing system.

36. (Previously Presented) The channel driver of claim 22, wherein the channel driver is operable to instantiate at least one driver object, wherein the at least one driver object is operable to interface with communication devices for different types of media.

37. (Previously Presented) The channel driver of claim 36, wherein the at least one driver object is operable to instantiate a service object.

38. (Previously Presented) The channel driver of claim 37, further wherein each service object includes a task thread to listen for incoming events from a communication device.

39. (Previously Presented) The channel driver of claim 37, wherein the service object is operable to interface with a communication server, and further wherein the communication server is operable to interface with a queuing system.

40. (Previously Presented) The channel driver of claim 39, wherein the queuing system is operable to assign work items to agents.

41. (Previously Presented) The channel driver of claim 22, wherein the commands in the interface command definition are implemented in a data link library.

42. (Previously Presented) The channel driver of claim 41, wherein commands in the interface command definition are accessed with a function pointer to the data link library.

43. (Previously Presented) The channel driver of claim 22, further comprising a task thread operable to listen for incoming events.

44. (Previously Presented) The channel driver of claim 43, wherein the task thread is operable to invoke an event handling function when an event is detected.

45. (Currently Amended) A method of inter-module communication between at least one channel driver and a communication server, wherein the channel driver is operable to

interface with one or more communication devices, and further wherein two or more of the communication devices can use different media types, the method comprising:

defining using a command definition **to support communication between the at least one channel driver and the communication server**, wherein
said command definition comprises commands for interfacing the at least one channel driver with the communication server, **and**
said commands are independent of a media type of a communication device using the channel driver.

46. (Previously Presented) The method of claim 45, further comprising:
invoking a command to request a media type list.

47. (Previously Presented) The method of claim 45, further comprising:
invoking a command to request a command event list.

48. (Previously Presented) The method of claim 45, further comprising:
invoking a command to create a driver object.

49. (Previously Presented) The method of claim 45, further comprising:
invoking a command to request a service object.

50. (Previously Presented) The method of claim 45, further comprising:
invoking a command to release a driver object.

51. (Previously Presented) The method of claim 45, further comprising:
invoking a command to issue a notice when handling of an event is complete.

52. (Previously Presented) The method of claim 45, further comprising:
invoking a command to suspend a work item.

53. (Previously Presented) The method of claim 45, further comprising:
invoking a command to resume a work item.

54. (Previously Presented) The method of claim 45, further comprising: invoking a command to handle a queued event.
55. (Previously Presented) The method of claim 45, further comprising: invoking a command to cancel a queued event.
56. (Previously Presented) The method of claim 45, further comprising: interfacing the communication server with a queuing system.
57. (Previously Presented) The method of claim 45, further comprising: detecting incoming events from the communication devices.
58. (Previously Presented) The method of claim 45, further comprising: instantiating a task thread to detect incoming events from the communication devices.
59. (Previously Presented) The method of claim 45, further comprising: detecting an incoming event from one of the communication devices; and invoking a function to handle the event.
60. (Previously Presented) The method of claim 59, further comprising: queuing the event to a memory cache.
61. (Previously Presented) The method of claim 60, further comprising: indicating the arrival of the event.
62. (Previously Presented) The method of claim 61, further comprising: dequeuing the event out of the memory cache and processing the event.
63. (Previously Presented) A computer readable storage media comprising: computer instructions to implement the method of claim 45.

64. (Previously Presented) A signal in a carrier medium comprising:
computer instructions to implement the method of claim 45.

65. (Currently Amended) An apparatus for inter-module communication comprising:
means for defining using a command definition to support communication via a multi-channel, multi-media, communication queuing system, wherein
said command definition comprises commands for interfacing with [[all] the
multi-channel, multi-media, communication queuing system, and
said commands are independent of a media type of a communication channel
of the multi-channel, multi-media, communication queuing system;
and
means for issuing a command of the commands of the command definition to the
communication channel.

66. (Currently Amended) The apparatus of claim 65 further comprising means for
defining using at least one of the following channel driver commands for: requesting media
type lists and command event lists, creating driver objects, requesting service objects, and
releasing driver objects.

67. (Currently Amended) The apparatus of claim 65 further comprising means for
defining using at least one of the following service object commands for: releasing service
objects, issuing a notice when handling of an event is complete, invoking commands, releasing
work items, suspending work items, resuming work items, handling queued events, and
canceling queued events.

68. (Currently Amended) The apparatus of claim 65 further comprising means for
defining using at least one of the following client object commands for: starting a work item,
releasing work items, saving work item contexts, restoring work item contexts, serializing work
items, freeing work item storage, beginning batch processing, and ending batch processing.

69. (Currently Amended) An apparatus for inter-module communication between at
least one channel driver and a communication server, wherein the each channel driver is

operable to interface with one or more communication devices, and further wherein two or more of the communication devices can use different media types, the apparatus comprising:

means for defining using a command definition, wherein
said command definition comprises commands configured for interfacing the at
least one channel driver with the communication server, and
said commands are independent of a media type of one device of the
communication devices using a particular channel driver of the at
least one channel driver; and
means for issuing a command of the commands of the command definition to the
one device.

70. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to request a media type list.

71. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to request a command event list.

72. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to create a driver object.

73. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to request a service object.

74. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to release a driver object.

75. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to issue a notice when handling of an event is complete.

76. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to suspend a work item.

77. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to resume a work item.
78. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to handle a queued event.
79. (Previously Presented) The apparatus of claim 69, further comprising:
means for invoking a command to cancel a queued event.
80. (Previously Presented) The apparatus of claim 69, further comprising:
means for interfacing the communication server with a queuing system.
81. (Previously Presented) The apparatus of claim 69, further comprising:
means for detecting incoming events from the communication devices.
82. (Previously Presented) The apparatus of claim 69, further comprising:
means for instantiating a task thread to detect incoming events from the communication devices.
83. (Previously Presented) The apparatus of claim 69, further comprising:
means for detecting an incoming event from one of the communication devices; and
means for invoking a function to handle the event.
84. (Previously Presented) The apparatus of claim 83, further comprising:
means for queuing the event to a memory cache.
85. (Previously Presented) The apparatus of claim 84, further comprising:
means for indicating the arrival of the event.
86. (Previously Presented) The apparatus of claim 85, further comprising:
means for dequeuing the event out of the memory cache and processing the event.

87. (Previously Presented) The method of claim 5, further comprising: invoking a command to request a media type list.

88. (Previously Presented) The method of claim 5, further comprising: invoking a command to request a command event list.

89. (Previously Presented) The method of claim 5, further comprising: invoking a command to create a driver object.

90. (Previously Presented) The method of claim 5, further comprising: invoking a command to request a service object.

91. (Previously Presented) The method of claim 5, further comprising: invoking a command to release a driver object.

92. (Previously Presented) The method of claim 5, further comprising: invoking a command to issue a notice when handling of an event is complete.

93. (Previously Presented) The method of claim 5, further comprising: invoking a command to suspend a work item.

94. (Previously Presented) The method of claim 5, further comprising: invoking a command to resume a work item.

95. (Previously Presented) The method of claim 5, further comprising: invoking a command to handle a queued event.

96. (Previously Presented) The method of claim 5, further comprising: invoking a command to cancel a queued event.

97. (**Currently Amended**) The method of claim 5, further comprising: interfacing [[the]] a communication server with a queuing system.

98. **(Currently Amended)** The method of claim 5, further comprising: detecting [[the]] a communication server with a queuing system.

99. **(Currently Amended)** The method of claim 5, further comprising: instantiating a task thread to detect incoming events from the communication ~~devices~~ channel.

100. **(Currently Amended)** The method of claim 5, further comprising: detecting an incoming event from ~~one of~~ the communication ~~devices~~ channel; and invoking a function to handle the event.

101. **(Currently Amended)** The method of claim 100 22, further comprising: queuing the event to a memory cache.

102. **(Currently Amended)** The method of claim 101 23, further comprising: indicating the arrival of the event.

103. **(Currently Amended)** The method of claim 102 24, further comprising: dequeuing the event out of the memory cache and processing the event.